SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I1P103

Volume 13 Issue 1, 27-45, January 2026

Original Article

TL-BERT: An Anti-Phishing Model Based on Transfer Learning and Transformer Mechanisms for Protective Social Networking

Manoj Kumar Prabakaran[1], Abinaya Devi Chandrasekar[2], Santhi Selvaraj[3], Abinaya Pandiarajan[4]

[1]*Department of Artificial Intelligence and Data Science, Mepco Schlenk Engineering College Sivakasi, Virudhunagar, Tamil Nadu, India.*
[2,3,4]*Department of Computer Science and Engineering, Mepco Schlenk Engineering College Sivakasi, Virudhunagar, Tamil Nadu, India.*

[1]*Corresponding Author : manojkumarp@mepcoeng.ac.in*

Received: 03 November 2025 Revised: 05 December 2025 Accepted: 04 January 2026 Published: 14 January 2026

Abstract *- Cybercrimes are growing exponentially in the digital era, and hackers continue to devise sophisticated cyber threats to gain unauthorized access. Among them, phishing remains one of the most prevalent and deceptive techniques used to exploit unsuspecting users. Although various preventive measures have been proposed by researchers in the past few decades, phishers are consistently adopting innovative strategies by deploying different forms of phishing URLs and webpage contents that are highly complex to detect in a real-time scenario. To address this issue, this work proposes TL_BERT: An anti-phishing model that integrates Transfer Learning (TL) with the Bidirectional Encoder Representations from Transformers (BERT) architecture. The model employs TL-adapted Autoencoders for extracting URL-based features and applies the BERT model to capture HTML-based textual features of a website. Both features are concatenated and classified using a Deep neural Network Model. Experiments were conducted on the benchmark dataset ISCXURL2016 dataset, which contains 54300 URL samples. The results indicate that TL_BERT attains a detection accuracy of 99.08% with a false positive rate of 1.01%. The optimized selection of lightweight architectures makes the proposed model a suitable entity for real-time deployment.*

Keywords *- Bidirectional Encoder Representations from Transformers, Hypertext Markup Language, Phishing detection, Transfer Learning, Uniform Resource Locator.*

1. Introduction

Phishing is a type of cyber-attack in which hackers develop illegitimate websites with the malicious intent of luring internet users into providing their valuable digital assets. In general, Uniform Resource Locators (URLs) of those websites are circulated through email across the internet society. A naive user who might not be able to discriminate between a real and fake website might fall into the trap of entering their private credentials, which might result in substantial economic and personal loss [1]. Over the years, hackers have come up with advanced strategies, such as domain name typo squatting or cybersquatting, in which the URL being crafted might look almost as close as possible to a real-world legitimate URL of a famous entity [2]. As per the report generated by Anti-Phishing Working Group (APWG), an international consortium that collects phishing-related fraudulent information, around 9,32,923 phishing attacks have been observed during the third quarter of 2024 alone. The most frequently targeted sector seems to be social media platforms, contributing 30.5% of all kinds of phishing attacks [3]. Initially, a few large-scale organizations, such as eBay, adopted the idea of blacklisting to mitigate phishing attacks, in which those sites that were identified as unsafe by internet users were recorded and displayed for safety purposes as blacklisted websites [4]. However, since the number of phishing URLs is growing at a rapid pace and phishers are generating a dynamic set of phishing websites, it would be practically impossible to detect the majority of the phishing websites using a blacklist due to its static nature [5].

In order to combat the dynamic nature of phishing URLs, researchers in the recent past concentrated on deploying Machine Learning (ML) models to detect the nature of real-world websites [6]. For this purpose, ML models were trained and tested using benchmark datasets that comprise a large number of legitimate and phishing URLs. In this way, models get to understand the significant nature of URLs instead of just verifying the existence of a phishing URL in the database, as in the case of blacklisting. This technique overcomes the problem of zero-day attacks since ML models understand the

structure of URLs based on their statistical, lexical, and domain-based features [7]. Extensive research was carried out in the recent past to identify the optimal set of features in order to reduce the computational overhead. Although state-of-the-art anti-phishing frameworks based on ML algorithms [8-10] have been proposed by researchers in the recent era, there are few vulnerabilities associated with ML-based phishing website detection, namely a) Reliance on third-party assistance for website-based feature extraction, b) Static features extracted might not always reflect evolving real-world phishing URLs since the nature of the URLs is changing significantly. c) ML-based models are not suitable for massive datasets comprising millions of website URLs [11].

Hence, in order to overcome the issues prevailing in the existing ML-based solutions, researchers in the recent past incorporated a representation learning mechanism for URL feature extraction instead of manual feature engineering [12-15]. Adoption of an illustration learning mechanism for feature extraction using DL based approaches has significantly overcome the problems associated with manual feature extraction.

However, most of the research works incorporating representation learning mechanisms have adopted only URL-based feature extraction, and not many such works have been conducted considering the features relevant to the HTML contents of a webpage. In order to construct an optimal phishing detection framework, both the URL and HTML content of a website shall be taken into consideration since they might better reveal the nature of the website content.

Based on an analysis of recent literature, the following key research gaps are identified in existing anti-phishing methodologies:

a) Existing machine learning based phishing detection solutions rely on manual feature engineering techniques to extract intrinsic URL and HTML characteristics. However, this task is tedious since it relies significantly on third-party engineering experts for crafting webpage features. Also, the handcrafted features do not reflect the dynamic nature of the real-world phishing webpage.
b) Modern deep learning-based phishing detection solutions that incorporate representation learning mechanisms often adopt advanced Neural Network architectures that automatically extract intrinsic URL and HTML features. Although this eliminates the reliance on third-party assistance, the features obtained using such techniques cannot be inferred, and there is a considerable structural complexity overhead with respect to the deployed Neural Network architecture.
c) Most of the existing Deep Learning based anti-phishing solutions either focus on URL or HTML content of the webpage. Not many dedicated research solutions have been proposed that concentrate on both the URL and HTML content of the webpage for identifying the authenticity of the webpage.

The above-identified research gaps highlight the key limitations associated with the existing anti-phishing solutions and underline the significance of an optimal phishing detection framework that shall potentially detect phishing websites in a real-time environment.

Hence, this work focuses on building a lightweight and optimal phishing website detection framework that would consider the inherent features of both URL and HTML content of a website into account. The proposed framework, termed "TL-BERT," is constructed by combining the merits of two advanced AI techniques: Transfer Learning (TL) and pre-trained transformers. Specifically, a lightweight TL-Enabled AE (TL_AE) model that receives the trained weight parameters of a pre-trained VAE architecture is utilized for URL feature extraction, and a base form of BERT model referred to as $BERT_{BASE}$ architecture is adopted for HTML feature extraction. Both the URL and HTML feature vectors obtained from TL_AE and BERT models were then concatenated and fed as input to the DNN for training and evaluation.

The proposed model has been experimented with a benchmark dataset that comprises both legitimate and phishing URL samples that widely represent various forms of real-world URL samples. The experimental results suggest that the proposed model possesses the ability to detect real-world phishing URL samples accurately. TL_BERT significantly eliminates the overhead associated with the existing manual feature engineering process, as well as considers both the URL and HTML features of a website for phishing detection. Adoption of a transfer learning mechanism and pre-trained transformers for representation learning makes the model lightweight and more suitable for real-time deployment.

The following are the novel contributions of the proposed TL_BERT framework:

a) Adoption of an advanced transfer learning mechanism for automatic URL feature extraction. In particular, a dedicated autoencoder model has been utilized to capture high-level latent space representation of the URL features effectively.
b) Preprocessing intrinsic HTML text content by the adoption of a unique HTML preprocessing mechanism that involves web scraping, cleaning, splitting, and organizing the content.
c) Adoption of pre-trained BERT transformer architecture to automatically extract context-aware text embedding vectors. To optimize the overall framework, a lightweight $BERT_{BASE}$ model has been utilized.
d) Implementing a unique concatenation layer in order to combine the extracted URL and HTML features

In summary, the proposed framework examines both URL and HTML features of a website for the detection of phishing websites. The model is constructed with the intention of deploying it as a browser add-on application, and hence, suitable measures have been taken to keep it lightweight and at the same time optimal for real-time deployment.

The rest of the chapters are organized as follows: Section 2 explains the literature survey conducted with respect to recent state-of-the-art anti-phishing solutions proposed by researchers across the globe. Section 3 details the proposed TL-BERT framework with the materials and methods adopted to construct the framework. Section 4 provides a detailed overview of the experimentations conducted and the results observed with respect to various metrics. Section 5 provides the conclusion of the entire research work being carried out.

2. Related Works

Phishing website detection has been one of the prominent research categories due to the extensive growth of social engineering attacks and the ever-growing, complicated attack mechanisms. Phishing attacks allow hackers to easily trap internet users into entering a fake website and gaining unauthorized access to their valuable assets. Also, the deployment of phishing attacks shall be easily done through the propagation of fake URLs via email and SMS, which shall reach millions of users within a few minutes. Over the years, the growth of phishing attacks has doubled significantly, resulting in substantial economic and personal losses. Hence, to mitigate the impact of phishing attacks, a wide array of anti-phishing solutions has been proposed by researchers that shall be broadly categorized based on their detection and learning mechanisms.

Initially, researchers incorporated a static blacklisting based phishing detection mechanism that allows users to report a webpage based on its authenticity. This technique was succeeded by Machine learning based solutions that learns discriminative URL and HTML patterns of a website to detect the nature of the website. Most recently, advanced Deep Learning-based architectures have gained significance due to their ability to automatically extract intrinsic URL and HTML features without manual feature engineering.

This section provides a detailed review of the existing anti-phishing solutions, categorized into blacklisting, machine learning-based approaches, and representation learning oriented deep learning-based mechanisms.The strengths and limitations of each anti-phishing technique are thoroughly discussed to showcase the significant research gaps addressed by the proposed framework.

2.1. Phishing Website Detection using Blacklisting

This section discusses the various anti-phishing mechanisms proposed by researchers in the recent past in order to overcome the impact of phishing attacks in real-world scenarios. One such earlier attempt made was the adoption of the blacklisting technique, in which a list of malicious website URLs was collected and maintained through the usage of various tools and techniques, namely automated web crawlers, user reports, security research, etc. Those blacklisted URLs were frequently updated and acted as a fundamental source for phishing attack detection [4].

Due to the simplicity of the blacklisting mechanism, many organizations deployed anti-phishing applications based on the idea of blacklisting, in which the users of those applications are warned when they try to access a website that appears in the blacklisted database. In particular, blacklisting was implemented in two unique ways: a) Server-side Blacklisting and b) Client-side Blacklisting. In the case of server-side blacklisting, the inherent features of suspicious URLs were kept in the server. Hence, the client shall send a query to the server in order to identify the malicious nature of a website. Some of the popular existing server-side blacklisting tools are as follows: i) eBay toolbar [16] ii) Netcraft Toolbar [17] iii) Web of Trust (WOT) [18], iv) TrustBar.

Similarly, client-side blacklisting was adopted by top MNC companies such as Google, Microsoft, etc., in which the client keeps the list of malicious websites in its local database instead of maintaining it on the server. Some of the popular client-side blacklisting tools are as follows: i) Google Safe Browsing [19], ii) McAfee Site Advisor [20], iii) Microsoft Smart Screen Service [21], iv) Websense Threat Seeker Network [22],

These toolbars provide security against malicious phishing websites by verifying the requested URL with the set of blacklisted URLs available in the local database as well as on the server. Although blacklisting provides assistance in effectively detecting phishing websites, it is pretty impractical to maintain an updated list due to the growing number of new phishing websites generated by phishers. In particular, it took 12 hours for 47% to 83% of phishing URLs to appear on the phishing websites [23]. It is a significant delay since nearly 63% of the phishing websites might victimize many users within the initial couple of hours. Henceforth, blacklisting is always susceptible to zero-day attacks, which is considered to be a significant drawback.

2.2. Phishing Website Detection using Machine Learning Algorithms

Hence, in order to mitigate the drawbacks associated with blacklisting techniques, various researchers concentrated on deploying Machine Learning (ML) models for phishing website detection. In contrast to the blacklisting technique, ML-based approaches incorporated both URL and HTML-based feature extraction mechanisms that shall further be used to train and evaluate a model to classify the nature of real-world websites [24] effectively.

In relevance to phishing website detection based on URL features, many research works were carried out during the past decade that focused on two main components of a traditional URL, namely Lexical/Statistical and Domain-based feature sets [25].

One such significant work was proposed by Gupta et al [26] in which the ISCXURL-2016 dataset was adopted for experimentation. In this work, nine lexical website features that represent the structural aspect of a website URL were considered for training and evaluation. Among the different classifiers, Random Forest achieved the maximum detection accuracy of 99.57%.

A similar work was conducted by Sajjad et al [27] that primarily focused on the behaviors and qualities of website URL by analyzing the lexical and domain-based features of a phishing website. In particular, the proposed approach utilizes the following URL-based characteristics, namely protocol scheme, hostname, path area, entropy, suspicious words, etc., to analyze the nature of a particular website. Six different datasets were chosen for experimentation, and 30 URL-based features were considered for training and evaluating the adopted ML classifiers. The experimental outcome showcases the superiority of the proposed framework, which exhibited higher accuracies of 96.25% and 94.65% on Kaggle datasets, respectively.

Apart from experimenting with standalone ML classifiers, a unique approach was proposed by Abdul et al (10) in the recent past that proposed a hybrid LSD model combining three unique ML classifiers, namely Linear Regression, Support Vector Machine, and Decision trees. Around 11000 URLs are collected from the Kaggle dataset that comprises 33 unique URL attributes representing the structural and domain attributes of phishing websites. The proposed method achieved 98.12% accuracy, demonstrating the effectiveness of URL feature extraction and the hybrid mechanism.

In addition to URL feature extraction, various research works were conducted to detect phishing websites that relied on actual webpage content and Hyper Text Markup Language (HTML) based features.

One of the earlier and effective attempts to detect phishing websites by extracting HTML content-based traits was proposed by Jain and Gupta [28], which is an entirely client-side solution that extracts 12 HTML-based hyperlink-specific features that represent the characteristics of a webpage's content. Various Machine learning algorithms, namely SMO, Naive Bayes, Logistic regression, Random Forest, Support Vector Machine (SVM), etc., have been experimented with. Among the different classifiers, logistic regression achieved a maximum accuracy of 98.42%.

Apart from experimenting with hyperlink-based features, a noteworthy research work has been conducted by Adebowale et al [29] that proposed an intelligent web-phishing detection using the integrated features of Images, Frames, and text. The model achieves a maximum detection accuracy of 98.3%. Another significant study relevant to HTML-based phishing website detection was carried out by Purwanto et al [30], which adopts a parameter-free similarity measure that examines the HTML of webpages and computes their similarity with known phishing websites, in order to classify them. The experimental results have shown that the model achieved an AUC score of 98.68% and a 0.58% false positive rate, respectively.

In addition to implementing phishing website detection with respect to URL and HTML-based features separately, a few researchers in the recent past proposed anti-phishing frameworks that consider both the URL and HTML content of a webpage [31]. One of the earlier attempts combining both URL and HTML features of a webpage for phishing detection has been proposed by Yuan et al [32], which incorporated both lexical features of a URL and links and contents in its webpage. Experimentations were carried out with a number of machine learning models, among which the Deep Forest model exhibited the maximum accuracy of 98.3%.

Aljofey et al [33] proposed an anti-phishing approach similar to [32] that integrates URL, hyperlink information, and textual content of a website. Integration of features enhanced the detection accuracy of the adopted XGBoost classifier, resulting in 96.76% with a lower false positive rate of 1.39%.

As per the studies conducted with respect to phishing website detection implemented using machine learning, it shall be inferred that it is more effective in comparison to the traditional blacklisting techniques. However, there are certain inherent limitations associated with ML-based phishing website detection, namely reliance on engineering experts to generate handcrafted URL and HTML features, static feature sets that do not reflect the ever-growing dynamic nature of a phishing website, and the model's inability to handle a massive volume of datasets [11].

2.3. Phishing Website Detection using Deep learning Algorithms: Automatic Feature Extraction using Representation Learning Mechanism

In order to overcome the limitations associated with existing ML-based anti-phishing solutions, researchers came up with the idea of adopting a representation learning mechanism in which both URL and HTML features were automatically extracted without the reliance on human intervention. In order to achieve this, various researchers adopted traditional Deep Learning algorithms for feature extraction as well as classification.

Yang et al [34] proposed a multi-dimensional feature-driven anti-phishing solution that incorporates the idea of a representation learning mechanism for automatic URL feature extraction. In this work, two significant DL models, namely Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM), were incorporated to extract URL features of a website automatically. Along with those features, statistical URL, webpage code, and text features were combined and fed as input to the classifier. The model exhibited a superior detection accuracy of 98.99%.

A unique approach combining both the URL and HTML-based automatic feature extraction mechanism has been proposed by Opara et al [15], referred to as Web Phish, that accepts URL and HTML source code as input and vectorizes them using a tokenizer utility class. Both the URL and HTML embedding vectors were concatenated and passed through the convolutional layers for classification. The proposed model achieved an average accuracy of 98% for three Fully Connected layers.

Although the adoption of traditional deep learning models, such as CNN, VAE, LSTM, etc., possesses the ability to extract intrinsic URL and HTML features of a website automatically, they lack the aptitude to measure the importance of the extracted features [35]. In order to overcome this, a few researchers in the recent past started incorporating transformer-oriented attention mechanisms through which features are weighed such that the most important features are selected and fed to the classifiers for training and evaluation.

Xiao et al [36] proposed a novel deep learning network that combines the efficacy of CNN and the Multi-Head Self Attention Mechanism (MHSA) for effective detection of phishing websites. The experimental results proved that the adoption of an attention mechanism in the process of weighing the extracted feature vectors significantly enhanced the detection ability of the classifiers, displaying an overall accuracy of 99.84%.

A similar kind of approach referred to as “CCBLA” was proposed by Zhu et al [37] that adopted Bi-LSTM along with CNN and attention mechanism in order to perform phishing website detection. The proposed model proved to be a lightweight entity displaying accurate phishing URL detection with minimal time consumption.

As per the studies conducted, it shall be inferred that attention-enhanced Deep Learning Models provided better detection outcomes in comparison to traditional deep learning models; however, these models typically required training from scratch, and they might struggle to generalize across diverse phishing patterns. Henceforth, researchers began experimenting with pre-trained transformer models and their variants, which are already trained with a vast data corpus, with the intention of proposing anti-phishing solutions that offer high performance with minimal fine-tuning and strong transfer learning abilities.

Maneriker et al [38] proposed URLTran that adopts pretrained transformer architectures for improving phishing detection. In this work, an existing transformer architecture is fine-tuned and pre-trained using URL data. Also, publicly available pretrained models such as BERT and RoBERTa were fine-tuned on the URL classification task. A cloze-style masked language modeling objective is applied to the BERT architecture. Each of these fine-tuned pretrained transformers was analyzed, and the best among them was finally selected as URLTran_BERT. The proposed URLTran produces an actual positive rate of 86.80% which is quite optimal.

Haynes et al [39] proposed a lightweight URL-based phishing detection using two state-of-the-art pre-trained transformers, namely BERT and ELECTRA. The experimental results confirm that the adopted deep transformers performed exceptionally well, producing detection accuracy values of 96.1% and 96.3% respectively. This work signifies the fact that the adoption of pre-trained transformers for phishing website detection enhances the detection outcome as well as minimizes the amount of time taken for training and validation.

Apart from individually adopting pre-trained transformers for phishing detection, an integrated approach combining deep learning classifiers along with pre-trained transformers has been proposed recently by Do et al [40], referred to as RasNet-TCMA-MPNet. In this work, in order to analyze the unique nature of phishing website URLs, RasNet combines keras embeddings with ResNet. In addition, Temporal Convolutional Neural Network (TCN) has been incorporated along with Multi-Head Self-Attention for the optimization of feature extraction. Also, in order to examine the natural language structure of webpage URLs, MPNet has been utilized. The proposed model exhibited a maximum detection accuracy of 99.71%. However, this work only focuses on the URL part of a phishing website and does not concentrate on the features associated with the webpage content.

3. Materials and Methods

In order to address the aforementioned limitations, an optimal lightweight phishing detection framework has been proposed that integrates the idea of transfer learning with a transformer-based architecture that concentrates on both the URL and HTML features of a phishing website. The proposed TL-BERT framework employs a TL-Enabled Autoencoder (TL-AE) model to automatically extract inherent raw URL features and Bidirectional Encoder Representation from Transformers (BERT) to generate context-aware HTML text embeddings. Both the URL and HTML features obtained from the TL-enabled AE and BERT models were concatenated and then fed as input to a traditional Deep Neural Network (DNN)

model for training and evaluation. Figure 1 depicts the architecture of the proposed TL-BERT model.

The proposed model is constructed by integrating two intelligent approaches for feature extraction and a neural network model for classification:

i) A traditional Auto Encoder (AE) model constructed based on a transfer learning mechanism that is dedicated to automatic feature extraction from preprocessed raw URL inputs.

ii) Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based deep learning model deployed for extracting valuable features from preprocessed HTML-based web content.

3.1. URL Feature Extraction using Autoencoders

According to a study conducted by [41], it can be inferred that AE models are well-suited for dimensionality reduction and feature extraction processes. Hence, considering the reconstruction ability of the AE model, a special form of AE architecture has been adopted, which incorporates transfer learning techniques to automatically extract raw URL features.

3.1.1. URL Preprocessing

The input URLs, which are in the form of text data, have been converted into fixed-size numerical vectors based on the One Hot Encoding (OHE) mechanism. OHE converts every character in a URL string into a fixed-size vector representation. For each character in the URL string, a vector is produced with respect to [42], which states that a Uniform Resource Locator (URL) of a website shall be formed based on 84 unique characters, including alphabets, digits, and special symbols. Hence, a corpus is generated with those 84 characters, each individually indexed.

For every character in the URL, an OHE vector of size 84 is generated, which contains 83 '0' values, and based on the index position with respect to the corpus, a '1' value is generated in accordance. Since neural network models expect the input vectors to be of uniform size, i.e., all URLs should have equivalent N * M dimensions, where N denotes the length of the URL and M represents feature representation size, based on [43], the M value is set to 84 uniformly. In contrast, N is determined by calculating the average length of all the URLs in the input dataset.

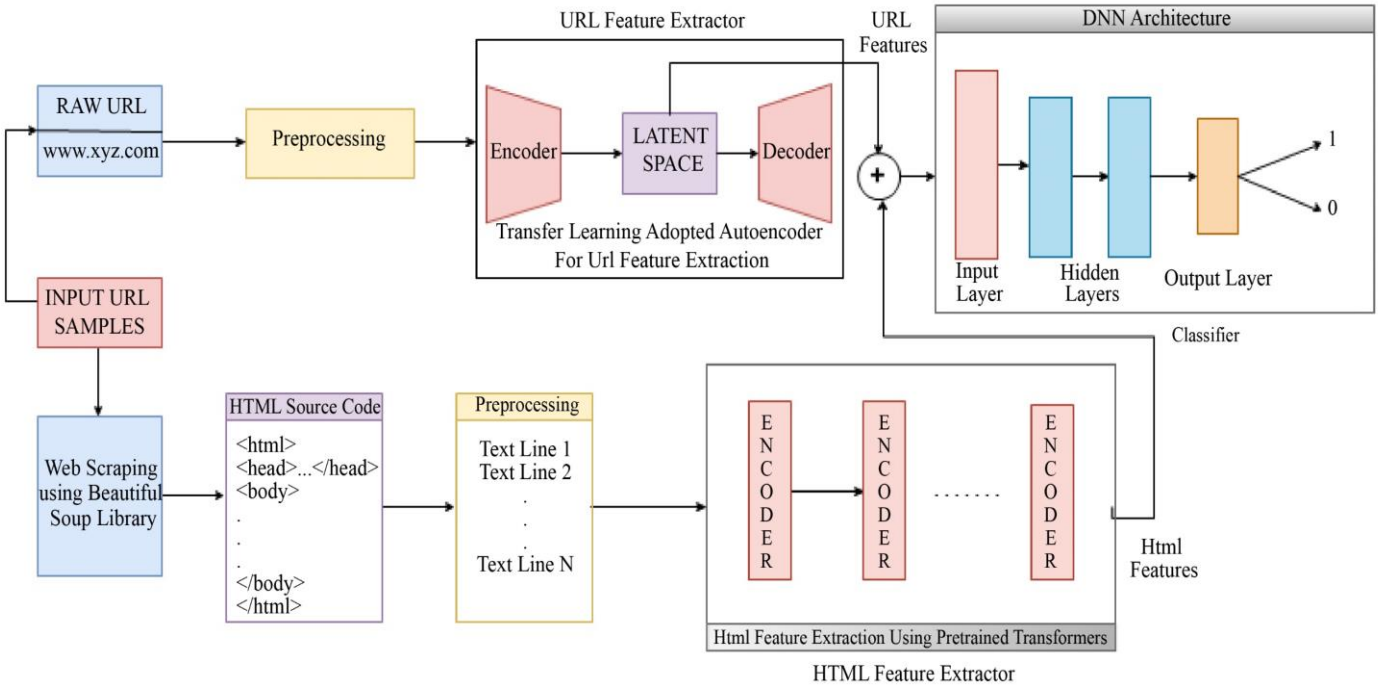


Fig. 1 Proposed TL-BERT architecture

Around 50000 URLs are available in the dataset, of which the average lengths of all the URLs were found to be 72. Hence, the N value is fixed to be 72 in this work. For those URLs whose length is shorter than 72, the remaining positions of the vector will be padded with 0s, and for those whose length exceeds 72, trimming is done to trim off those extra characters in the URL. Finally, after preprocessing the input URLs, a numerical vector of dimension N *M is obtained, where N = 72 and M = 84, respectively.

3.1.2. Autoencoders as Feature Extractors

Figure 2 depicts the detailed workflow of the proposed feature extraction module. The preprocessed numerical vectors of individual URLs are collectively fed as input to the

feature extraction module. As already suggested, a traditional AE model is employed for the process of URL feature extraction. An autoencoder model [44] is a variant of Neural Network Architecture that encompasses three main layers: a) encoder, b) latent space, and c) decoder. The basic principle behind the AE model is to receive input data and produce an output that is a reconstructed form of the input data.

The AE model is structured by stacking three layers sequentially, namely the encoder layer, the latent space, and the decoder layer. The encoder part of the AE model is structured as follows: It consists of an input layer, three hidden layers, and an output layer. The number of units in the input layer of the encoder is set to be 84, since it should match the preprocessed URL vector dimension.

Hidden layer 1 consists of 80 neuron units, followed by hidden layer 2 containing 70 units and hidden layer 3 holding 64 units, respectively. The number of units in the latent space layer of AE is set to be 20. The decoder part of AE is constructed in the reverse manner with respect to the encoder part of the model, as shown in Figure 2.

The primary steps involved in training an AE model are as follows:

1. The preprocessed URL vectors are fed as input to the encoder layer, where the data gets forwarded to the intermediate hidden layers and finally to the output layer.
2. The output of the encoder part is a dimensionally reduced input vector, which gets stored in the latent space or bottleneck layer of the architecture.
3. The units in the latent space region are then forwarded as input to the decoder part of the model. Here, the vectors are decompressed by passing through a set of hidden layers to the output layer.
4. The output layer produces a URL vector that is of the same dimension as the original input vector.
5. Reconstruction error is calculated based on the difference between the input vector and the reconstructed output vector. Based on the loss value obtained, the weights of the decoder and encoder are adjusted during the backpropagation process.
6. Once the reconstruction loss is optimally minimized, the training process is halted.

After the completion of training, the dimensionally reduced latent space vectors shall be extracted and provided as one of the inputs to the classification layer in the TL-BERT model. The latent space vectors are taken into account since they contain dimensionally reduced abstract higher-level representations of the input data.

However, instead of training an AE model using a bunch of real-world URLs from scratch, a special technique is adopted that eases the training process of the model, such that valiant features are extracted at less computational cost and in a limited time duration.

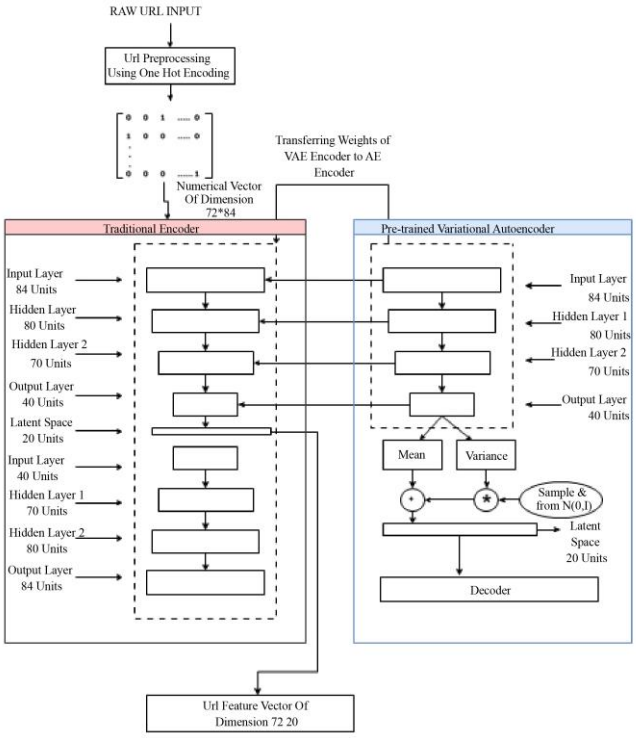


Fig. 2 Detailed workflow of the feature extractor module

In order to speed up the training duration of the model, an advanced machine learning technique referred to as Transfer Learning (TL) is adopted. In TL [45], a pre-trained model that has been exposed to a massive dataset for a unique task shall be used as the initiating point to train a particular model. Instead of making use of one of the existing pre-trained models, a self-built Neural Network Model that has already been trained with a vast number of real-world URLs is incorporated as an initiator to train the AE architecture. The architecture utilizes a Variational Autoencoder (VAE) framework. Specifically, the encoder weights from the pre-trained VAE model described in are transferred to the AE encoder in this study to leverage previously optimized feature extraction capabilities. VAE model, in particular, was trained with a custom dataset consisting of 1 48 960 URL samples, out of which 74,480 are real-world phishing URLs and the remaining 74,480 are generated adversarial phishing samples. For generating the adversarial URL samples, a special form of generative modeling technique referred to as Generative Adversarial Network (GAN) is implemented. The GAN model was being trained for around 5000 epochs to generate fake phishing samples, which in turn were combined with real-world phishing URLs collected from benchmark datasets to train the VAE architecture.

Considering the merits of the constructed VAE model with respect to the number of training samples and its

exposure to adversarial URL inputs, a transfer learning mechanism is adopted in which the encoder part of the VAE architecture is completely transferred, along with the updated weights to the proposed AE model.

The encoder part of the VAE model, comprising an input layer, three hidden layers, an output layer, and the latent space layer, is entirely replicated along with the updated weights in the AE architecture. For training the TL-enabled AE model, around 50,000 URL samples are collected from benchmark datasets. The number of epochs is set to be 50. The learning rate is fixed at 0.001. The Adam optimizer is used as the optimization technique. The mean square error is used as the loss function to calculate the reconstruction error. The encoder part is frozen for the initial 35 epochs, and only the decoder part undergoes back propagation, i.e., the weights are adjusted only for the decoder layer units initially. For the remaining 15 epochs, both the encoder and decoder units undergo weight updates.

3.2. HTML Feature Extraction using Pre-Trained Transformers

Numerous existing solutions [28-30] in relevance to HTML-based phishing website detection have been proposed by various researchers in the recent past, which dictated the significant role of HTML features in the classification outcome. Considering the merits of the automatic HTML feature extraction technique as well as the complexities associated with representation learning, a transformer-based model is deployed to automatically extract context-based text embedding of webpage content.

3.2.1. HTML Preprocessing

The following steps are performed to preprocess the webpage content into a clean format before being fed to the transformer model for feature extraction.

- Initially, web scraping is done by implementing BeautifulSoup, a Python library that shall be imported from the BS4 package. This enables us to parse HTML documents in a user-friendly manner.
- Once the HTML documents are parsed, the source code content of the respective webpage will be retrieved. From the source code, script and style tags are ripped off, and the relevant text contents are extracted alone using the get_text () method associated with the BS library.
- After extracting the text content, those texts are broken into separate lines by removing leading and trailing spaces on each sentence and further split into individual chunks.
- Finally, a dictionary is generated with a key as URL and values in the form of a nested list comprising line-by-line text extracted from chunks using the splitlines () method.
- In order to make each entry in the dictionary of uniform length, suitable threshold values, say ‘T are identified that set the fixed length for all the webpage text content. Based on the experimentation conducted, the T value was fixed to be 500. Hence, those website contents whose text length exceeded the T value were trimmed off, and those having a length shorter than the T value were padded with additional 0’s for compensation.

Once the preprocessing stage is finished, a dictionary is obtained with key and value pairs, where key denotes the individual URLs in the dataset and values represent the preprocessed webpage content organized in the form of a multi-dimensional list with uniform length.

To extract the intrinsic features of a webpage context with respect to contextual embedding, this work incorporates BERT (Bidirectional Encoder Representations from Transformers). This pre-trained language model uses the Transformer architecture [46]. In particular, $BERT_{BASE,}$ a pre-trained model that comprises 12 encoders stacked together linearly, has been implemented. The encoder part of the model constitutes a feed-forward network layer containing 768 neuron units and 12 attention heads.

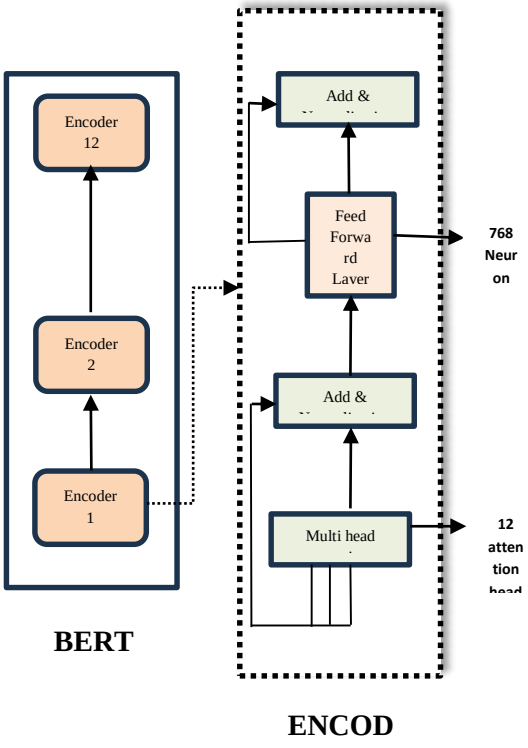


Fig. 3 $BERT_{BASE}$ architecture

The working principle of a BERT model is as follows:

- Initially, the model receives input in the form of text tokens. Here, each word or sentence in a text shall be considered an individual token.
- The first input token for any form of input is a [CLS] token that refers to classification.

- Followed by the input token, either a sequence of words or a sentence, depending on the requirement, shall be fed as input to the model, which would flow up through the stack of encoders in the respective BERT model.
- In each layer, there are 'n' numbers of self-attention heads that apply a self-attention mechanism to calculate the attention weights of each token and further pass the attention score calculated through the FFN layer.
- The FFN layer finally hands over the obtained output data to the next encoder in the stack.
- Each input token produces an output vector of a different size depending on the BERT variant.

Figure 3 depicts the architecture of the adopted $BERT_{BASE}$ along with the internal structure of the encoder part associated with the model.

The $BERT_{BASE}$ model has been implemented using the TensorFlow library, which is available in Python. The model has been downloaded from the following URL: https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-512_A-8 , and a respective preprocessing URL that indulges in text preprocessing has been downloaded from https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3. Algorithm 1 explains the steps involved in extracting HTML features using the BERT model.

Figure 4 represents the detailed representation of HTML feature extraction using the adopted BERT model. Initially, the preprocessed HTML data, which is in the form of a dictionary, is fed as input to the $BERT_{BASE}$ model for feature extraction. The dictionary contains a key-value pair where the key points to a particular URL in the input dataset and the value denotes the preprocessed textual web content organized as a multi-dimensional list.

The keys in the dictionary are iterated, and the text values of a particular URL are fed line by line as input to the BERT model. This model comprises a sequence of 12 encoders. The preprocessed HTML feature vectors were passed sequentially onto all 12 encoders with a CLS and SEP tag at the initial layer. For each line, a contextual embedding vector of size 512 is generated.

Since the length of the webpage content 'T' was fixed uniformly as 500, an output vector of dimension 500 * 512 will be generated for each website URL available in the input dataset. The obtained HTML features based on the BERT model play a vital role in phishing website detection since those features are based on the context in which a particular text is present.

Algorithm 1 HTML feature extraction using BERT

```
    from bs4 import BeautifulSoup
    html = fetch_html_from_url(url)
    soup = BeautifulSoup(html, 'html.parser')
    text_content = soup.get_text()
    cleaned_content=
    remove_unwanted_elements(text_content)
    lines = split_into_lines(cleaned_content)
    data_dict = {url: lines}
    return data_dict
    //Generating feature vectors using BERT
    HTML_vectors=[ ]
    for key in data_dict:
     output_vector = []
for layer in bert_model.encoders:
    attention_weights = self_attention_mechanism(token,
    layer)
    attention_scores = ffn_layer(attention_weights)
output = attention_scores
output_vector.append(output)
HTML_vectors.append(output_vector)
```

3.3. Deep Neural Network (DNN) Model for Classification

The final layer of the proposed TL-BERT architecture incorporates a Deep Neural Network (DNN) model that is tasked with classifying legitimate and phishing websites. DNN receives its input from the feature vectors extracted from both the AE and BERT models. Both the URL and HTML feature vectors were concatenated and produced as a combined numerical vector for the DNN model for URL prediction.

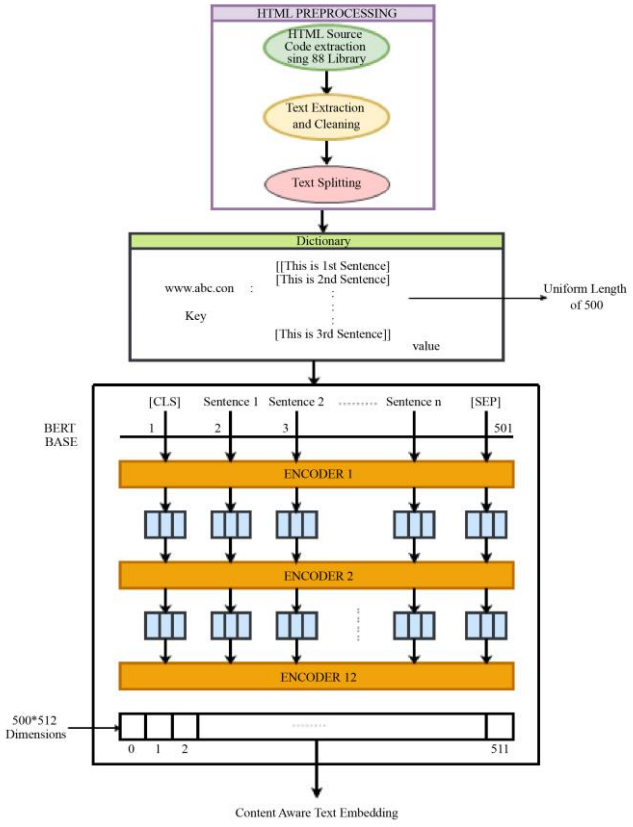


Fig. 4 HTML feature extraction using BERT

The extracted URL feature vector is of dimension N * M, where N = 72 and M = 20, respectively. HTML feature vectors obtained through the BERT model are of dimension L * Z, where L = 500 and Z = 512. Before feeding these features into the classifier for prediction, they need to be flattened into a single input vector of dimension [Ui, Xi, Yi], where Ui denotes a particular URL in the input dataset, Xi is calculated as 572, which is obtained by adding N and L values. In contrast, Yi is set to 532 by summing up M and Z values.

The flattened vector [Ui, Xi, Yi] shall be produced as input to the proposed DNN model for URL classification. The adopted DNN model is structured as follows: It consists of an input layer, four hidden layers, and an output layer. The number of units in the input layer is fixed at 532, corresponding to the dimension of the input URL vector. Four hidden layers comprise 512, 256, 128, and 64 neuron units. The output layer contains two units for binary classification purposes. Figure 5 depicts the detailed representation of the classification layer.

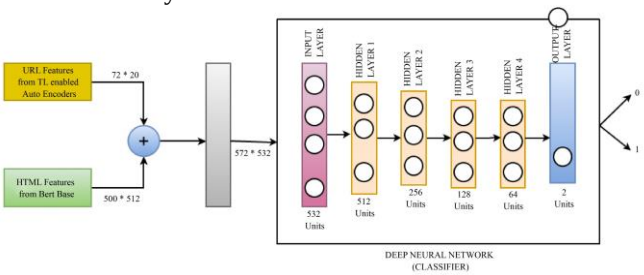


Fig. 5 Classification layer

Initially, the input dataset is split into training and testing samples. The ratio of training and testing data was fixed at 80:20. The DNN model undergoes supervised training in which it receives a combined URL and HTML features along with its label values. The number of training epochs was set to 50, and the learning rate of the model was fixed at 0.001.

The input vectors were passed through the set of hidden layers and finally to the output layer. The loss value is estimated by calculating the difference between the predicted and actual output. A negative log-likelihood loss function has been adopted to estimate the loss value. Batch normalization was done based on a dropout mechanism. Furthermore, to prevent overfitting, a dropout mechanism is implemented, whereby specific neuron units are randomly dropped during the training process. Also, to ensure optimal training, the batch size was fixed at 100. The Adam optimizer was used to optimize the loss function. At every intermediate layer, the Rectified Linear Unit (ReLU) was used as the activation function, except for the output layer, in which the Log-Softmax function was used as an activation function for calculating the probabilistic outcome.

Finally, to ensure the efficient training of DNN without complex computations, the concept of early stopping was implemented in order to cease the training process whenever the validation loss stopped improving. Once the training is completed, the model is exposed to the testing data samples for detecting malicious URL samples.

4. Experimental Results and Analysis

4.1. Dataset Description

Raw URL samples have been collected from the following two benchmark datasets, namely Alexa Top Website and ISCX URL 2016, for the purpose of training the proposed TL-BERT model. The dataset comprises 54300 URL samples extracted from the above-mentioned benchmark resources. The constructed dataset is a balanced mixture of legitimate and phishing URL samples.

The legitimate URL samples were crawled from Alexa (Source: www.alexa.com), a web traffic analysis company owned by Amazon that provides information and rankings on the popularity of websites. It determines website rankings based on factors such as daily unique visitors, page views, and average time spent on the site. Also, it provides a list of the top websites globally and for specific countries, along with additional analytics and insights. Exactly 29870 URLs were crawled from the Alexa website during the time period of March 2023. Phishing samples were collected from the ISCX URL 2016 dataset (Source: https://www.unb.ca/cic/datasets/url-2016.html), which comprises four variants of phishing URLs, namely spam, phishing, and defacement URLs. Out of these 4 variants, 24430 phishing URLs were randomly collected to form the list of phishing URLs in the dataset.

The dataset was split into training and testing data, of which 70% of the input data was considered for training purposes and the remaining 30% for testing. A total of 16290 URL samples were used for evaluating the proposed model, of which 8961 were benign, and 7329 were malicious.

In order to accurately assess the performance of the proposed model, various measures were taken into consideration, namely the accuracy curve, loss curve, confusion matrix, precision, recall, F1 score, True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), False Negative Rate (FNR), and Area Under the ROC (AUC-ROC) curve.

The experimental setup comprises Google Colab Pro, a cloud-based Jupyter notebook environment, which provides a seamless and setup-free platform for conducting the experiments. Google Colab Pro offers several advantages, including access to powerful hardware resources and GPU acceleration. Specifically, the experiments made use of the Tesla V100 PCIe GPU accelerator, which is known for its high performance in deep learning tasks. This GPU has a staggering 14 TFLOPS (Tera Floating-Point Operations Per Second) of computational power, allowing for efficient and fast model

training and evaluation. Additionally, the Tesla V100 boasts a substantial memory bandwidth of 900 GB/sec, enabling smooth data transfer and processing. To complement the powerful GPU, Google Colab Pro provided generous system specifications. The setup included a sizable 125GB HDD, which allowed for ample storage of datasets, model checkpoints, and experimental outputs. Moreover, the system provided 25GB of memory, ensuring sufficient space for running memory-intensive tasks and accommodating large-scale models and datasets.

4.2. Design of Experimentation and Result Analysis

Experiments were conducted in different phases to analyze the importance of both URL and HTML features in the effective classification of malicious URLs. In each phase, the impact of URL and HTML features with respect to classifier accuracy was analyzed by experimenting with different feature extraction strategies. The following are the three phases in which the experiments were carried out:

- Impact analysis of the proposed TL-BERT model with respect to URL features alone (Phase 1).
- Impact analysis of the proposed TL-BERT model with respect to HTML features alone (Phase 2).
- Impact analysis of the proposed TL-BERT model with respect to both the URL and HTML features (Phase 3).
- Lightweightness evaluation of the proposed TL_BERT model (Phase 4)
- Evaluation of the proposed TL-BERT model against the existing state-of-the-art anti-phishing approaches that employed representation learning techniques (Phase 5).

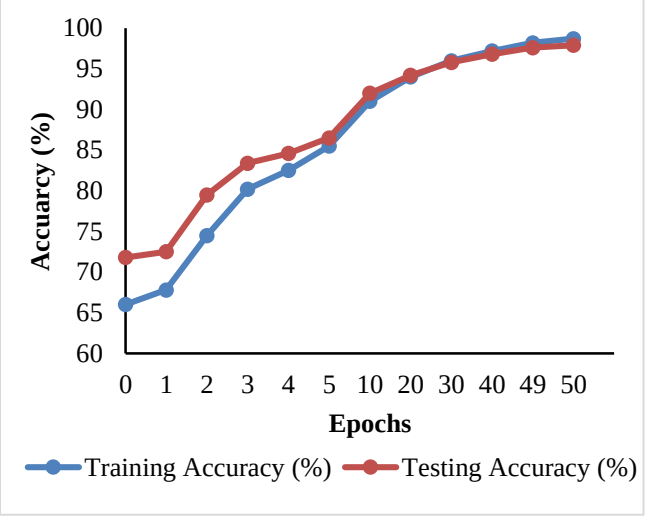
4.2.1. Phase 1- URL Feature Analysis

In this phase, the HTML feature extractor module of the proposed framework is excluded, and only the URL feature extractor and classifier are considered for experimentation.

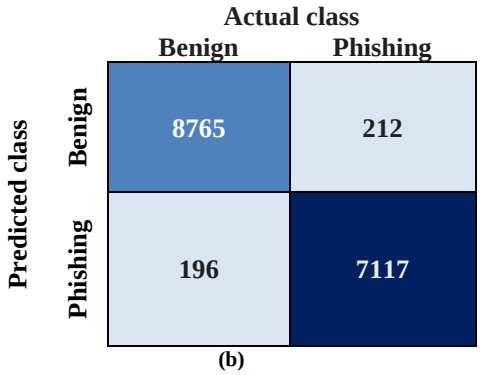
Hence, minor alterations have been made with respect to the dimensions of the features extracted from the TL-assisted AE model. The classifier's input layer is composed of 532 units, which were purposefully fixed with respect to the combined input size of both URL and HTML features. Since only URL features are dealt with alone in this experiment, which are of dimension L * 20 (L specifies the uniform length of a particular URL), it is mandatory to align the URL feature size with the classifier's input layer size. Hence, a padding mechanism is adopted for compensation in which the required number of zeroes is padded to produce a URL feature vector of dimension L*532. Although padding is applied, the inherent features extracted by the extractor remain unchanged. These features were then fed as input to the classifier for training and evaluation.

In this experimental phase, the TL-assisted AE model is taken into account without changing its structure with respect to the original proposed TL-BERT model. The performance of the model designed with respect to URL features alone has been measured using the following two metrics: accuracy curve and confusion matrix.

Figure 6 (a) and 6 (b) represent the accuracy curve and confusion matrix for the experimented TL-enabled AE-DNN model. As can be inferred from the results, the model reaches a maximum accuracy of 97.5%. The number of false positives acquired was 212 out of the 7329 malicious URL samples, and the number of false negatives obtained was 196 out of the 8961 benign samples.



(a)



(b)

Fig. 6 (a) Accuracy curve of TL-Enabled AE-DNN, (b) Confusion matrix of TL-Enabled AE-DNN.

The model exhibits a better outcome with the URL features alone, and this is mainly due to the adoption of transfer learning techniques in the feature extraction process. In order to stress the effect of adopting transfer learning in URL feature extraction, a separate experiment was carried out in which the role of TL was excluded from the picture. In this process, the traditional autoencoder was trained from scratch without leveraging the trained weight initialization technique. In the TL-disabled AE architecture, both the encoder and decoder parts of the AE model undergo weight updates during the entire training process. No weights were transferred from

any pre-trained models. Once the training process is completed, the latent space feature vectors are fed as input to the DNN for URL classification.

The performance analysis of the features extracted with respect to the traditional AE model is measured based on a similar metric adopted for the TL-enabled AE model. Figure 7(a) and 7(b) depict the accuracy curve and confusion matrix for the TL-disabled AE-DNN model.

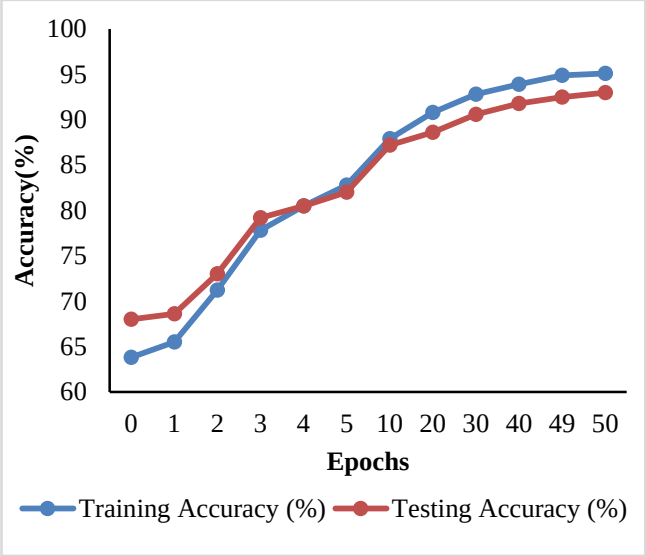


Fig. 7 (a) Accuracy curve of TL-Disabled AE-DNN, (b) Confusion matrix of TL-Disabled AE-DNN.

As per the results obtained, it can be found that the detection accuracy of the model reaches a maximum of 92.5%. However, the number of false positive and false negative samples was significantly higher when compared with the previous case. The model acquires a false positive rate of 6.49%, which is quite large.

4.2.2. Phase 2- HTML Feature Analysis

In the proposed TL-BERT architecture, along with the HTML embedding vector, the URL feature vectors were concatenated to produce the input. Instead, in this experimental phase, the URL features were ignored for experimental purposes, and only the BERT-generated embedding vector of dimension 500*512 was taken into account for analysis. Since the input layer of the DNN model comprises 532 neuron units, a padding mechanism is adopted in the HTML embedding vector to compensate for the size of the DNN input layer. Hence, the additional 20 spaces are allocated to the BERT embeddings and populated with 0s.

Apart from adopting the BERT text embedding mechanism, there are various other real-world text embedding techniques that can convert real-world text data into fixed-size numerical vectors. Hence, in this experimental phase, five different text embedding mechanisms were experimented with apart from BERT to assess the quality of the text-based features obtained with respect to individual techniques in the effective identification of phishing websites, namely Term Frequency – Inverse Document Frequency. (TF-IDF), Bag of Words (BoW), Global Vector For Word representation (Glove), Word2Vec, and Fast Text [47]. Figure 8 describes the detailed performance analysis of each text embedding technique with respect to precision, recall, F1 score, and accuracy metrics.

From the experimental results obtained, it can be observed that out of all the text embedding techniques, the adopted BERT architecture exhibits the best outcome in terms of precision, recall, and F1 score. BERT reaches a maximum F1 score of 0.98, which is the highest among all the other text embedding techniques. This clearly suggests that context-aware embedding vectors play a significant role in describing the nature of the content of a particular website. Apart from BERT, glove-generated embedding vectors lead to a better detection accuracy of 96.1%, which is the second highest among all the experimented models. However, the precision and recall values obtained for the glove model with respect to phishing samples were not optimal.

In contrast, the TF-IDF-based feature vector delivers the least detection accuracy of 91.5% with a lower precision and recall value in comparison to other techniques. Both Word2Vec and FastText produce similar kinds of results, exhibiting an average F1 score of 0.95. Table 1 summarizes the performance outcomes of the various text embedding techniques.

Table 1. Performance outcome of various text embedding techniques

Text embedding techniques	Precision	Recall	F1-score	Accuracy (%)
TF-IDF	0.9135	0.9153	0.9143	91.49
BoW	0.9501	0.9470	0.9483	94.90
Word2Vec	0.9507	0.9501	0.9504	95.10
FastText	0.9568	0.9562	0.9565	95.70
Glove	0.9607	0.9603	0.9605	96.10
BERT	0.9807	0.9808	0.9809	98.10

The following are the inferences obtained after experimenting with different text embedding techniques: a) TF-IDF does not capture the semantic relationships between words and may struggle to handle out-of-vocabulary words. b) Both Word2Vec and FastText were memory-intensive and consumed more data to train. c) Glove is effective at capturing semantic relationships between words and can handle out-of-vocabulary words. However, the drawback associated with Glove is that it is memory-intensive and requires a significant amount of data. d) BERT captures the contextual meaning of words by considering the entire input sentence, leading to better representation of word meanings in natural language processing tasks.

	Precision	Recall	F1-score
Benign	**0.932321**	0.911617	0.921853
Phishing	**0.894793**	0.919089	0.906778
Accuracy	0.914979	0.914979	0.914979
Macro avg	0.913557	0.915353	0.914315
Weighted avg	0.915437	0.914979	0.915071
	Performance outcome of TF-IDF + DNN		

(a)

	Precision	Recall	F1-score
Benign	0.941656	**0.967191**	**0.954253**
Phishing	**0.95851**	**0.926729**	0.942352
Accuracy	0.948987	0.948987	0.948987
Macro avg	0.950083	0.94696	0.948302
Weighted avg	0.949239	0.948987	0.948898
	Performance outcome of BoW + DNN		

(b)

	Precision	Recall	F1-score
Benign	0.95838	**0.96362**	**0.960993**
Phishing	0.95522	**0.948833**	0.952016
Accuracy	0.956967	0.956967	0.956967
Macro avg	0.9568	0.956227	0.956504
Weighted avg	0.956958	0.956967	0.956954
	Performance outcome of FastText + DNN		

(c)

	Precision	Recall	F1-score
Benign	0.953042	**0.95804**	**0.955535**
Phishing	0.948366	**0.942284**	0.945315
Accuracy	0.950952	0.950952	0.950952
Macro avg	0.950704	0.950162	0.950425
Weighted avg	0.950938	0.950952	0.950937
	Performance outcome of Word2Vec + DNN		

(d)

	Precision	Recall	F1-score
Benign	0.96286	**0.966298**	**0.964576**
Phishing	0.958613	**0.954428**	0.956516
Accuracy	0.960958	0.960958	0.960958
Macro avg	0.960737	0.960363	0.960546
Weighted avg	0.960949	0.960958	0.96095
	Performance outcome of Glove + DNN		

(e)

	Precision	Recall	F1-score
Benign	0.982918	**0.98248**	**0.982699**
Phishing	**0.97859**	0.979124	0.978857
Accuracy	0.98097	0.98097	0.98097
Macro avg	0.980754	0.980802	0.980778
Weighted avg	0.980971	0.98097	0.98097
	Performance outcome of BERT + DNN		

(f)

Fig. 8 Precision, recall, F1 score, and accuracy of various text embedding techniques. Panels show, (a) TF-IDF, (b) BoW, (c) FastText, (d) Word2Vec, (e) GloVe, and (f) BERT.

In a nutshell, it can be summarized that the adoption of BERT to generate context-aware embedding of HTML text content results in optimal phishing website detection in comparison to all the other text embedding mechanisms.

4.2.3. Phase 3- Analysis of Proposed TL-BERT Model

The core objective of this research is to deploy an AI-assisted phishing website detection mechanism that optimally identifies malicious URLs in a real-world environment. To enhance the detection ability of the proposed model, this research intends to enrich the detector with both the URL and HTML features of a particular website. In the previous two phases of experimental analysis, URL and HTML features were individually experimented with to study their impact on the model's classification ability. In this phase, the core task is to analyse the proposed TL-BERT model, which actually combines both URL and HTML features obtained by employing the TL-enabled AE and BERT models. The following experiments have been carried out in phase 3 of the experimental analysis:

- Initially, the proposed TL-BERT model is experimented with the input dataset, and the detection ability of the model is assessed using accuracy and a loss curve.
- To further showcase the optimality of the TL-BERT architecture, additional experiments were conducted to compare the performance outcomes of the model with those of the models constructed in the previous two phases (TL-AE DNN and BERT + DNN). The outcomes were analysed using the following metrics: TPR, TNR, FPR, and FNR.

- In order to validate the combined effect of URL and HTML in website classification, a study is conducted by constructing an AuC-ROC curve that examines the classification ability of the model with respect to URL, HTML, and URL+HTML.

Figure 9 demonstrates the accuracy and loss curve obtained for the proposed TL-BERT model, and the results clearly suggest that the proposed model performed predominantly well, reaching a maximum accuracy of 99.08%. Although the model struggled to classify the URLs in the initial epochs of the training process, it can be witnessed that after a very few epochs, the model adapted in accordance. This is mainly due to the adoption of pre-trained models to extract URL and HTML features.

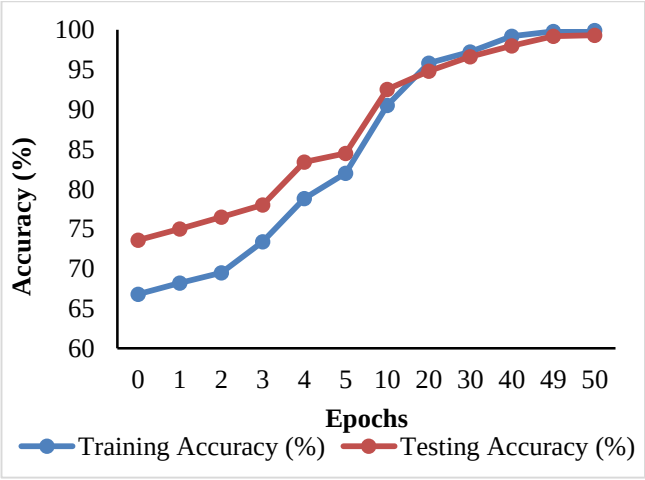
Additional experiments were conducted to evaluate the performance of the proposed model with respect to the models constructed in the previous two phases. Table 2 presents the performance analysis of the models in all three phases, specifically with respect to TPR, TNR, FPR, and FNR.

Table 2. Performance analysis of TL-BERT with respect to phase 1 and phase 2 models

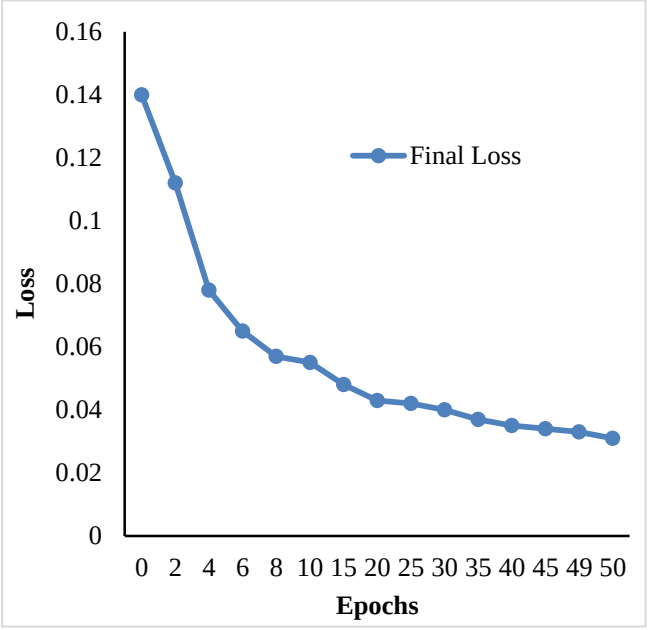
Proposed models	True Positive Rate (TPR)	False Positive Rate (FPR)	True Negative Rate (TNR)	False Negative Rate (FNR)
TL-AE + DNN	97.81%	2.89%	97.11%	2.19%
BERT + DNN	98.25%	2.09%	97.91%	1.75%
TL-BERT+DNN	99.15%	1.01%	98.99%	0.85%

With respect to the URL features (TL-AE+DNN), although the classifier exhibits a decent TPR and TNR value, there is a significant increase in the FPR value, and this is mainly because focusing only on URL features might lead the model to misclassify specific malicious URLs as legitimate ones. As can be observed, the TL-AE model reached an FPR of 2.89%, which is the highest among the experimented models.

In the case of HTML features (BERT+DNN), the performance outcome was better when compared with the URL features. The model exhibits lower FPR and FNR values of 2.09 and 1.75%, respectively. This outcome is achieved on the basis of the context-aware embedding mechanism that leads the classifier to understand the contextual information of HTML text content to precisely identify malicious websites. As can be witnessed from Table 2, the proposed TL-BERT architecture displayed an excellent outcome in terms of correctly identifying benign and malicious URLs. The model acquired the lowest FPR and FNR values of 1.01% and 0.85%, which is considered to be quite decent for the quantity of URLs taken for evaluation.



(a)



(b)

Fig. 9 (a) Accuracy curve of the TL-BERT Model, (b) Loss curve of TL-BERT Model.

Figure 10 shows the AuC-ROC curve of the proposed model for both TL-AE and BERT. This analysis was done to ensure the trade-off between sensitivity and specificity of the experimental models.

The ROC curve clearly suggests that the proposed TL-BERT model delivers the maximum AuC score of 0.98, which is almost closer to 1. This shows the ability of the model to effectively classify benign and malicious websites.

Combining the merits of both URL and HTML features results in effective identification of URLs with a lower false positive rate and higher detection accuracy.

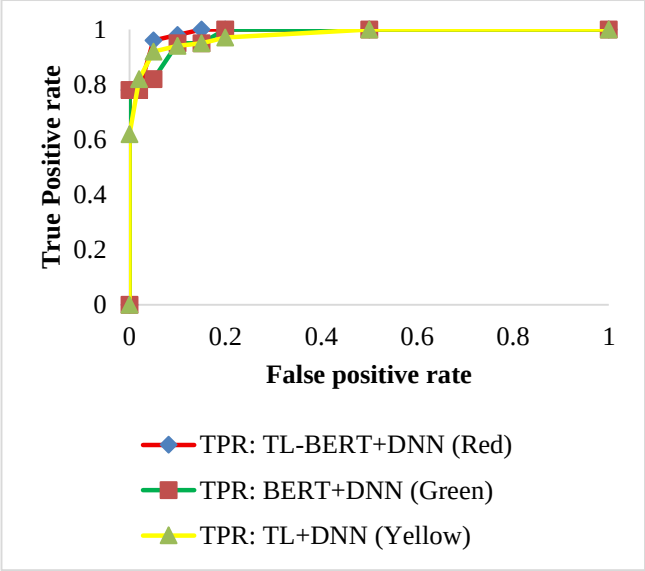


Fig. 10 AuC-ROC curve of the experimented models

4.2.4. Phase 4- Light-Weightedness Evaluation of the Proposed TL_BERT Model

In this phase, a special experiment has been conducted to analyze the optimality and lightweightedness associated with the proposed model. In order to evaluate the model, the following metrics have been chosen, namely Training time/epoch, Inference time/sample, memory usage, and detection accuracy.

For performing a fair evaluation, the following similar kinds of architectures have been derived that are slightly modified versions of the proposed TL_BERT architecture. The following are those modified phishing detection architectures constructed for this experimentation:

a) VAE + $BERT_{base}$: In this configuration, instead of the TL-adopted AE model, a pretrained VAE architecture is employed to initialize the encoder, which is then integrated with the BERTbase model. However, $BERT_{base}$ is utilized in the proposed model.
b) AE(w/o TL) + $BERT_{base}$: A similar design to TL_BERT except that both the encoder and decoder part of the AE model is trained from scratch without the adoption of a transfer learning mechanism
c) VAE + $BERT_{large}$: This structure involves the combination of Variational autoencoders and $BERT_{large,}$ a special form of BERT architecture that produces 1024 embedding vectors.
d) AE (w/o TL) + $BERT_{large}$: In this case, $BERT_{large,}$ along with traditional autoencoders implemented without a TL mechanism, is fused

All of the above constructed models are evaluated against the proposed TL_BERT architecture. In particular, DNN is deputed as the classifier with a similar configuration as TL_BERT for all the experimented models in this phase. The core objective of this experimentation is to assess the lightweightedness of the model, which shall be measured with respect to training time and inference, as well as the optimality that is validated using detection accuracy. Table 3 shows the Runtime and resource efficiency comparison of TL_BERT with the experimentally structurally modified phishing detection framework.

Table 3. Runtime and resource efficiency comparison of TL_BERT framework

Model variant	Training time/ Epoch (seconds)	Inference time/sample (milliseconds)	Memory usage	Accuracy (%)
VAE + $BERT_{base}$	65	24	1.1 GB	98.72
AE(w/o TL) + $BERT_{base}$	58	21	950 MB	98.31
VAE + $BERT_{large}$	95	39	2.5 GB	98.88
AE(w/o TL) + $BERT_{base}$	78	33	2.3 GB	98.42
Proposed TL_BERT model	**42**	**18**	**900 MB**	**99.08**

As can be observed from Table 3, the proposed model consumed the minimal training time duration of 42 seconds and an inference time of 18 milliseconds, which is the most optimal among all the experimented models.

Also, the amount of memory consumed by each architecture significantly differed according to its structural complexities, with vast amounts of memory being consumed by two architectures that adopted $BERT_{large,}$ averaging around 2.4 GB. The least amount of memory has been utilized by the model, consuming 900MB respectively. With lower training time, faster inference speed, and optimal memory usage, this model exhibited a significant advantage for real-time deployment. In addition to the lightweight nature of TL_BERT, TL_BERT achieved the maximum detection accuracy of 99.08% outperforming all the experimented variants.

This experimental analysis concludes that the proposed model is both lightweight and optimal and is well-suited for real-time deployment.

4.2.5. Phase 5 – Comparison of TL-BERT with Current Anti-Phishing Solutions

In this phase, the proposed TL-BERT model has been compared against the current state-of-the-art anti-phishing approaches. In particular, unique anti-phishing models that deploy representation learning techniques to identify malicious URLs have been identified for this experimentation. In fact, the proposed TL-BERT model incorporates a representation learning technique in which both URL and HTML features are automatically extracted, excluding the process of manual feature engineering. Hence, it would be more appropriate to compare the proposed model with those approaches that deploy automatic feature extraction mechanisms. Seven significant phishing detection solutions proposed in the recent past have been considered for comparative evaluation.

Table 4 demonstrates the details of those approaches along with their performance analysis. The results projected in Table 4 were in accordance with the evaluation values as provided in the respective papers.

Table 4. Comparison of TL-BERT with current anti-phishing methods

State-of-the-art Phishing detection approaches	Feature set	Precision (%)	Recall (%)	F1 score (%)	Accuracy (%)
PDRCNN	URL	97.33	93.78	95.52	95.6
HTML Phish	HTML	97	98	97	98
WebPhish	URL and HTML	98	98	98	98
PhishDet	URL and HTML	96.40	96.44	96.42	96.42
Web2Vec	URL, HTML, and DOM Structure	98.69	98.26	98.47	99.05
MFPD	URL and HTML	99.41	98.57	99	98.88
Proposed TL_BERT model	**URL and HTML**	**99.06**	**99.07**	**99.07**	**99.08**

Precise Phishing Detection with Recurrent Convolutional Neural Network (PDRCNN) [48] is an anti-phishing technique that employs bi-directional LSTM and CNN, which in particular rely only on the URL of a particular website. Although it is a faster and lighter mechanism, it exhibits an average recall value of 93.78%. HTML Phish [49] is a CNN-based phishing webpage classification technique that relies only on the HTML webpage content for classification.

Out of the seven research works, the following solutions, WebPhish [12], Phish Det [15], and Multi-dimensional Features driven by Deep learning (MFPD) [34], adopted both URL and HTML features for phishing website detection. All of those approaches employed different means to automatically extract URL and HTML features in order to identify malicious websites optimally. In particular, MFPD outperforms all the other models, reaching a maximum accuracy of 98.88% and a higher F1 score of 99%. This is mainly because the model adopted multi-dimensional URL and web page features that are both statistical and automatically driven. The only limitation of MFPD is the model's reliance on manually crafted URL and HTML statistical features, along with other extracted features. Web2Vec [14] is a phishing website detection model that not only focuses on URL and HTML features but also considers the DOM structure of the webpage, which makes the model quite expensive. The model is hybrid in nature, combining the merits of CNN and bi-directional LSTM, which tends to produce the highest accuracy among the experimented models, reaching a value of 99.05%.

The proposed TL-BERT model achieves a maximum accuracy of 99.08% with a notable F1 score. Compared to other models, the proposed model is lightweight and less computationally intensive. Notably, the model leverages the training overhead associated with other models since only pre-trained models for feature extraction have been incorporated.

4.3. Discussion

The performance of the TL_BERT framework in comparison to the existing state-of-the-art solutions can be attributed to its effective adoption of advanced transfer learning and pre-trained transformer-based contextual learning. Unlike the existing traditional machine learning solutions that implement manual feature engineering, the proposed model leverages a transfer learning adopted autoencoder and a pre-trained BERT model to automatically extract significant URL and HTML features of a phishing website. In comparison to the modern deep learning based solutions, namely CNN-LSTM, CNN-GRU, and other hybrid frameworks, TL-BERT exhibits better detection outcomes in terms of precision, recall, and accuracy metrices. The core reason behind the stability of TL_BERT lies in the underlying structure of the proposed framework, which comprises two

lightweight infrastructures: a transfer learning-adopted lightweight autoencoder model and a base version of a pre-trained BERT model. This makes the model lightweight and optimal, making it less prone to structural complexities and huge training time overhead. This highlights the dominance of the proposed TL_BERT with respect to the existing anti-phishing solutions, making it much more suitable for real-time deployment in an attack-prone environment.

5. Conclusion

The core objective is to build a lightweight, optimal model that accurately detects malicious websites with minimal false alarm rates. In order to meet the fundamental objective, a transfer learning enabled autoencoder model has been constructed for the role of automatic URL extraction. The choice of adopting the TL mechanism is to minimize the complexity associated with training the AE model for URL feature extraction. Since the weight parameters of a pre-trained VAE architecture were transferred to the proposed traditional AE model, only the decoder part will undergo training. Hence, the complexity associated with training the AE for feature extraction is eliminated, leading to a lightweight infrastructure. Also, for the role of extracting HTML features from the website, a transformer-based BERT model is adopted that generates fixed-size context-aware text embedding vectors through a post-preprocessing mechanism. However, to reduce the complexity associated with this process, $BERT_{BASE,}$ a special variant of the BERT architecture, is adopted, which is a lightweight architecture capable of generating 512 vectors. The choice of $BERT_{BASE}$ instead of a $BERT_{LARGE}$ architecture is to reduce the overall structural complexity of the proposed framework. Both the generated URL and HTML feature vectors were concatenated and given to the output layer for classification. This technique of adopting a based AE model and a base variant of BERT architecture helps us to ensure that TL-BERT remains lightweight while maintaining high detection accuracy. The proposed experimental results show that the TL-BERT framework achieved a maximum accuracy of 99.08% with a 1.01% false rate. The training time associated with both URL and HTML feature extraction modules was significantly reduced due to the use of transfer learning and pre-trained models, making the model faster and lighter.

Additionally, the core objective of this research is to build browser-based add-on software, in which the proposed TL-BERT model will be deployed for real-time detection of phishing websites. Once deployed, the model's ability shall be periodically analyzed by setting up a feedback loop that logs the details of misclassified results. This mechanism enables us to continuously monitor the model's performance after deployment, facilitating further training and fine-tuning.

Acknowledgments

The authors would like to thank Mepco Schlenk Engineering College for their valuable support and encouragement throughout this research work.

References

[1] Ike Vayansky, and Sathish Kumar, "Phishing – Challenges and Solutions," *Computer Fraud & Security*, vol. 2018, no. 1, pp. 15-20, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[2] Janos Szurdi et al., "The Long "Taile" of Typosquatting Domain Names," *Proceedings of the 23rd USENIX Security Symposium*, San Diego, CA, pp. 191-206, 2014. [Google Scholar] [Publisher Link]

[3] Anti-Phishing Working Group, "*Phishing Activity Trends Report, 3rd Quarter 2024*," Unifying the Global Response to Cybercrime, pp. 1-11, 2024. [Publisher Link]

[4] Tara Baniya, Dipesh Gautam, and Yoohwan Kim, "Safeguarding Web Surfing with URL Blacklisting," *2015 12th International Conference on Information Technology - New Generations*, Las Vegas, NV, USA, pp. 157-162, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[5] Steve Sheng et al., "An Empirical Analysis of Phishing Blacklists," *Proceedings of the 6th Conference on Email and Anti-spam (CEAS)*, Mountain View, California USA, pp. 1-10, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[6] Ammar Odeh, Ismail Keshta, and Eman Abdelfattah, "Machine Learning Techniques for Detection of Website Phishing: A Review for Promises and Challenges," *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, NV, USA, pp. 813-818, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Shamshair Ali et al., "Comparative Evaluation of AI-Based Techniques for Zero-Day Attacks Detection," *Electronics*, vol. 11, no. 23, pp. 1-25, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Manu J. Pillai et al., "Evasion Attacks and Defense Mechanisms for Machine Learning-Based Web Phishing Classifiers," *IEEE Access*, vol. 12, pp. 19375-19387, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Naya Nagy et al., "Phishing URLs Detection Using Sequential and Parallel ML Techniques: Comparative Analysis," *Sensors*, vol. 23, no. 7, pp. 1-17, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] Abdul Karim et al., "Phishing Detection System through Hybrid Machine Learning Based on URL," *IEEE Access*, vol. 11, pp. 36805-36822, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Alsharif Abuadbba et al., "Towards Web Phishing Detection Limitations and Mitigation," *arXiv Preprint*, pp. 1-12, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Subhash Ariyadasa, Shantha Fernando, and Subha Fernando, "Combining Long-Term Recurrent Convolutional and Graph Convolutional Networks to Detect Phishing Sites Using URL and HTML," *IEEE Access*, vol. 10, pp. 82355-82375, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Chenguang Wang, and Yuanyuan Chen, "TCURL: Exploring Hybrid Transformer and Convolutional Neural Network on Phishing URL Detection," *Knowledge-Based Systems*, vol. 258, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Subhash Ariyadasa, Subha Fernando, and Shantha Fernando, "Detecting Phishing Attacks Using a Combined Model of LSTM and CNN," *International Journal of Advanced and Applied Sciences*, vol. 7, no. 7, pp. 56-67, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] Chidimma Opara, Yingke Chen, and Bo Wei, "Look Before You Leap: Detecting Phishing Web Pages by Exploiting Raw URL and HTML Characteristics," *Expert Systems with Applications*, vol. 236, pp. 1-13, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Stay safe on eBay, eBay. [Online]. Available: https://pages.ebay.com/securitycenter/

[17] Netcraft Anti-Phishing Toolbar, Netcraft. [Online]. Available: https://toolbar.netcraft.com

[18] WOT: Web of Trust – Website Reputation and Security, Web of Trust. [Online]. Available: https://www.mywot. com

[19] Google Safe Browsing: Protecting Users from Phishing and Malware, Google Security Blog. [Online]. Available: https://safebrowsing.google.com

[20] McAfee SiteAdvisor: Website Safety Ratings and Security Analysis, McAfee Security. [Online]. Available: https://www.mcafee.com/en-in/safe-browser/mcafee-webadvisor.html

[21] Microsoft Defender SmartScreen: Protection against Phishing and Malware, Microsoft Security. [Online]. Available: https://learn.microsoft.com/en-us/windows/security/operating-system-security/virus-and-threat-protection/microsoft-defender-smartscreen/

[22] Forcepoint ThreatSeeker, Forcepoint. [Online]. Available: https://www.forcepoint.com/product/feature/threatseeker

[23] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones, "Phishing Detection: A Literature Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2091-2121, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[24] Lizhen Tang, and Qusay H. Mahmoud, "A Survey of Machine Learning-Based Solutions for Phishing Website Detection," *Machine Learning and Knowledge Extraction*, vol. 3, no. 3, pp. 672-694, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[25] Doyen Sahoo, Chenghao Liu, and Steven C.H. Hoi, "Malicious URL Detection using Machine Learning: A Survey," *arXiv Preprint*, pp. 1-37, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[26] Brij B. Gupta et al., "A Novel Approach for Phishing URLs Detection Using Lexical Based Machine Learning in a Real-Time Environment," *Computer Communications*, vol. 175, pp. 47-57, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[27] Sajjad Jalil, Muhammad Usman, and Alvis Fong, "Highly Accurate Phishing URL Detection Based on Machine Learning," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 7, pp. 9233-9251, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[28] Ankit Kumar Jain, and B.B. Gupta, "A Machine Learning Based Approach for Phishing Detection Using Hyperlinks Information," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 2015-2028, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[29] M.A. Adebowale et al., "Intelligent Web-phishing Detection and Protection Scheme using Integrated Features of Images, Frames and Text, Frames and Text," *Expert Systems with Applications*, vol. 115, pp. 300-313, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[30] Rizka Widyarini Purwanto et al., "PhishSim: Aiding Phishing Website Detection with a Feature-Free Tool," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1497-1512, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[31] Sultan Asiri et al., "A Survey of Intelligent Detection Designs of HTML URL Phishing Attacks," *IEEE Access*, vol. 11, pp. 6421-6443, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[32] Huaping Yuan et al., "Detecting Phishing Websites and Targets Based on URLs and Webpage Links," *2018 24th International Conference on Pattern Recognition (ICPR)*, Beijing, China, pp. 3669-3674, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[33] Ali Aljofey et al., "An Effective Detection Approach for Phishing Websites Using URL and HTML Features," *Scientific Reports*, vol. 12, pp. 1-19, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[34] Peng Yang, Guangzhen Zhao, and Peng Zeng, "Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning," *IEEE Access*, vol. 7, pp. 15196-15209, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[35] Wenhao Li et al., "A State-of-the-Art Review on Phishing Website Detection Techniques," *IEEE Access*, vol. 12, pp. 187976-188012, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[36] Xi Xiao et al., "CNN–MHSA: A Convolutional Neural Network and Multi-Head Self-Attention Combined Approach for Detecting Phishing Websites," *Neural Networks*, vol. 125, pp. 303-312, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[37] Erzhou Zhu et al., "CCBLA: A Lightweight Phishing Detection Model Based on CNN, BiLSTM, and Attention Mechanism," *Cognitive Computation*, vol. 15, pp. 1320-1333, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[38] Pranav Maneriker et al., "URLTran: Improving Phishing URL Detection Using Transformers," *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, San Diego, CA, USA, pp. 197-204, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[39] Katherine Haynes, Hossein Shirazi, and Indrakshi Ray, "Lightweight URL-Based Phishing Detection Using Natural Language Processing Transformers for Mobile Devices," *Procedia Computer Science*, vol. 191, pp. 127-134, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[40] Nguyet Quang Do et al., "An Integrated Model Based on Deep Learning Classifiers and Pre-Trained Transformer for Phishing URL Detection," *Future Generation Computer Systems*, vol. 161, pp. 269-285, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[41] Mayu Sakurada, and Takehisa Yairi, "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction," *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, Gold Coast Australia QLD Australia, pp. 4-11, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[42] T. Berners-Lee, L. Masinter, and M. McCahill, "*Uniform Resource Locators (URL)*," IETF RFC 1738, pp. 1-25, 1994. [CrossRef] [Google Scholar] [Publisher Link]

[43] Allen Chieng Hoon Choong, and Nung Kion Lee, "Evaluation of Convolutionary Neural Networks Modeling of DNA Sequences Using Ordinal Versus One-Hot Encoding Method," *2017 International Conference on Computer and Drone Applications (IConDA)*, Kuching, Malaysia, pp. 60-65, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[44] Dor Bank, Noam Koenigstein, and Raja Giryes, *Autoencoders*, Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook, pp. 353-374, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[45] Shuteng Niu et al, "A Decade Survey of Transfer Learning (2010–2020)," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 2, pp. 151-166, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[46] Jacob Devlin et al., "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Minneapolis, Minnesota, pp. 4171-4186, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[47] Mustafa Nabeel Salim, and Ban Shareef Mustafa, "A Survey on Word Representation in Natural Language Processing," *AIP Conference Proceedings: 1st Samarra International Conference for Pure And Applied Sciences*, Samarra, Iraq, vol. 2394, no. 1, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[48] Weiping Wang et al., "PDRCNN: Precise Phishing Detection with Recurrent Convolutional Neural Networks," *Security and Communication Networks*, vol. 2019, pp. 1-15, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[49] Chidimma Opara, Bo Wei, and Yingke Chen, "HTMLPhish: Enabling Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis," *2020 International Joint Conference on Neural Networks (IJCNN)*, Glasgow, UK, pp. 1-8, 2020. [CrossRef] [Google Scholar] [Publisher Link]